New Parallel Routines for Dense Complex Eigenvalue Applications

Mark R. Fahey
Computational Migration Group
Computer Sciences Corporation
U.S. Army Engineer Research and Development Center
Major Shared Resource Center
Vicksburg, MS 39180

Two new codes for computing the eigenvalues and eigenvectors of a complex Hessenberg matrix are presented. The first computes the Schur decomposition of a complex Hessenberg matrix, while the second computes the eigenvectors of a complex upper triangular matrix. These subprograms have been developed to fill a void in the ScaLAPACK library. Now, the capability exists to compute the eigenvalues and eigenvectors of dense complex matrices using a parallel QR algorithm.

A parallel complex Schur decomposition routine was developed based on the real Schur decomposition routine provided in ScaLAPACK. The real code was appropriately modified to make it work with complex arithmetic as well as to make the algorithm implement a complex multiple bulge QR algorithm. This also required the development of new auxiliary routines that perform essential operations in the complex Schur decomposition, and that will provide additional linear algebra computation capability to the ScaLAPACK community.

A parallel eigenvector calculation routine was also developed. This routine can take the output from the parallel Schur decomposition subprogram and compute the eigenvectors for the original complex Hessenberg matrix.

This work also presents the parallelization of two application codes that require the eigenvalues and eigenvectors of dense complex matrices. The first is from the

area of electromagnetics. In particular, the application calculates radiation from a long antenna situated on a photonic crystal substrate. Most of the computational effort involves repetitious calculation of eigenvalues and eigenvectors. The new parallel solvers significantly reduce the wallclock time.

The second application is from the area of magnetohydrodynamics in the study of systems with combined flow and magnetic shears. For example, this application is used to study the formation and acceleration of slow solar wind and the accompanying coronal streamer magnetic field. This requires the calculation of eigenvalues of dense complex matrices. In serial, as above, more than 90% of the computational effort is spent computing eigenvalues.

List of Contributors

J. Merle Elson
Signals and Sensors Division
Naval Air Warfare Center

R. Dahlburg
Laboratory for Computational Physics
and Fluid Dynamics
Naval Research Laboratory

1 Introduction

In this work, two new parallel codes for computing the eigenvalues and eigenvectors of a complex Hessenberg matrix are discussed. The first computes the Schur decomposition of a complex Hessenberg matrix, while the second computes the left and/or right eigenvectors of a complex upper triangular matrix.

The conversion of the ScaLAPACK [3] code `PDLAHQR` to a complex implementation is covered. `PDLAHQR` computes the Schur decomposition of a real matrix in parallel using a multiple bulge QR algorithm. Therefore, this new implementation, called `PZLAHQR`, computes the Schur decomposition of a complex matrix in parallel using a multiple bulge strategy. In addition, a parallel code was developed to compute the left and/or right eigenvectors of an upper triangular matrix. This code is denoted as `PZTREVC`. Since the output from `PZLAHQR` is standard Schur form, `PZLAHQR` and `PZTREVC` can be used in conjunction with existing ScaLAPACK routines to compute eigenvalues and eigenvectors of a general complex matrix in parallel. The names of these new codes follow the ScaLAPACK convention.

Furthermore, the parallelization of two application codes with the new solvers that require the computation of eigenvalues and eigenvectors of complex dense matrices is presented. The first is from the area of electromagnetics, while the second is from the area of magnetohydrodynamics. The new parallel solvers have significantly reduced the wallclock time for runs of each code.

In Section 2, highlights of the parallel nonsymmetric QR algorithm are presented from [12]. The conversion of the real parallel nonsymmetric QR algorithm to complex arithmetic is discussed in Section 3. In Section 4, a parallel complex eigenvector routine is briefly presented. Scalability results for `PZLAHQR` and two case studies of converting a serial code to parallel with ScaLAPACK and the new solver routines discussed here are presented in Section 5. Concluding remarks are given in Section 6.

2 Parallel QR Algorithm

A parallel nonsymmetric QR algorithm for real matrices was implemented in the code `PDLAHQR` as part of ScaLAPACK. The algorithm used in `PDLAHQR` is similar to the LAPACK [1] routine `DLAHQR`. However, unlike `DLAHQR`, instead of sending one double-shift through the largest unreduced submatrix, this algorithm sends multiple double-shifts. This allows all bulges to carry out up-to-date shifts and spaces them apart so that there can be parallelism across several processor row/columns. Another critical difference is that this algorithm applies multiple double-shifts in a block fashion, as opposed to `DLAHQR`, which applies one double-shift at a time.

This is the approach taken in [12] where M shifts are obtained from the lower $M \times M$ submatrix, where M is a fairly large even number (say 40), and used to form $S = M/2$ bulges of degree two and chase them one after the other down the subdiagonal in parallel. Key observations pertaining to parallelization discussed by Henry, Watkins, and Dongarra [12] are as follows:

- The most critical difference between serial and parallel implementations of the QR algorithm is that the number of bulges must be chosen to keep the processors busy. The bulges must be separated by at least a block, and remain synchronized, to ensure that each row/column of processors remains busy. Usually the block size must be large; otherwise there will be too much boundary communication.

- The overall logic can be kept similar to the well-tested QR algorithm. The super-iteration can be implemented to complete before new shifts are determined and another super-iteration is begun. Information about the "current" unreduced submatrix must remain global to all nodes.

- The Householder transforms are of size 3, which means they are specified by sending 3 data items. The latency associated with sending many such small messages would be ruinous, so the information from several (e.g., 30) Householder transformations is bundled in each message.

- If many bulges are being chased simultaneously, there may be several bulges per row or column of processors. In that case, latency can be reduced further by combining the information from all bulges in a given row or column into a single message.

3 Conversion of `PDLAHQR` to `PZLAHQR`

The original need to develop a parallel complex Schur decomposition routine arose during the migration of a researcher's code from a scalar to parallel platform. The most time-consuming computational task in this code was computing the full eigen-decomposition of a dense complex matrix. Without the time or resources to develop an entire parallel code from start to finish, the decision was made to convert the existing ScaLAPACK code `PDLAHQR` to a complex implementation. This would have the drawback of not developing a parallel code based on the multiple single-shift strategy usually associated with complex QR algorithms. Counterparts to the auxiliary codes developed with `PDLAHQR` had to be developed, as well as developing a serial complex double-shift QR algorithm counterpart `ZLAHQR2` to the LAPACK code `DLAHQR`. LAPACK already has a routine named `ZLAHQR` that employs a single-shift strategy.

Some of the new routines needed in the development of `PZLAHQR` are listed next, each with a brief description. For more details, see [11].

`ZLAHQR2` Serial complex double-shift QR algorithm, implemented by converting the LAPACK single-shift routine `ZLAHQR` to a double-shift strategy. The double-shift strategy only reduces the Hessenberg matrix to quasi-triangular form with 2×2 blocks on the diagonal; thus these blocks must be further reduced to triangular form.

`ZLANV2` Computes the Schur decomposition of 2×2 blocks, i.e.,

$$\begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \bar{w} & \bar{x} \\ 0 & \bar{z} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

This routine computes a rotator matrix (composed of cosine c and sine s values), which needs to be applied to the corresponding rows and columns of the Hessenberg matrix.

PZROT Applies in parallel a rotator matrix to two rows or columns of a distributed matrix.

PZLAHQR Parallel multiple-bulge QR algorithm implemented by converting the ScaLAPACK routine PDLAHQR to complex format. This routine requires the use of ZLAHQR or ZLAHQR2, ZLANV2, and PZROT.

4 Eigenvector Calculation

PZLAHQR computes the Schur form of a complex Hessenberg matrix. Although the eigenvalues are on the diagonal of the upper triangular matrix, the Schur decomposition does not produce the eigenvectors. Thus, additional calculations must be performed to compute the eigenvectors of the upper triangular matrix produced by PZLAHQR. The eigenvector matrix can then be postmultiplied by the unitary matrix obtained in the Schur decomposition to give the eigenvectors of the original Hessenberg matrix.

Another parallel routine was developed, PZTREVC, that computes some or all left and/or right eigenvectors of a complex upper triangular matrix, i.e.,

$$Tx = \lambda x \qquad \text{and/or} \qquad y^H T = \lambda y^H$$

where T is upper triangular, x and y are right and left eigenvectors, respectively, and λ is an eigenvalue. Together, PZLAHQR and PZTREVC can be used to compute the eigenvalues and eigenvectors of a complex Hessenberg matrix. Since ScaLAPACK already has a routine to reduce a nonsymmetric complex matrix to Hessenberg form, one can now compute the eigendecomposition of a complex nonsymmetric matrix in parallel.

The right eigenvectors are computed via backward substitution using the PBLAS. A similar technique is used to compute the left eigenvectors using forward substitution routines from the PBLAS.

Note that this code may give inaccurate results if the eigenvectors are ill-conditioned. Furthermore, overflow may occur. Overflow can be controlled by the use of scaling. For PZTREVC, an auxiliary solver PZLATRS was implemented to control scaling in the computation of the eigenvectors. However, the parallel implementation of PZLATRS does not scale well and should only be used when necessary.

5 Numerical Results

Numerical tests were carried out on an SGI Origin 2000 and IBM Power3 SMP at the U.S. Army Engineer Research and Development Center (ERDC) Major Shared Resource Center (MSRC). See Table 1 for more information on each machine.

All solver routines are implemented in Fortran 77, except for PZROT, which is coded in C. All routines were compiled using optimization flags -O3 and -O3

Table 1. *Parallel computing platforms at the ERDC MSRC used for testing.*

Machine	Processor Speed (MHz)	Mflops/s per proc	Procs per node	Number Nodes	Peak Gflops/s
SGI Origin 2000	195	390	2	64	49.9
IBM Power3 SMP	222	888	8[a]	64	454.6

[a]Current hardware limitation of a maximum of four MPI processes per node.

`-qarch=pwr3 -qtune=pwr3 -qmaxmem=-1` for the SGI Origin 2000 and the IBM Power3 SMP, respectively. The environment variable `MP_SHARED_MEMORY` was set to `YES` for runs on the IBM Power3 SMP. All tests were run during normal operation hours in a nondedicated environment.

5.1 Scaled Problem Size Scalability of `PZLAHQR`

The scalability of `PZLAHQR` is first investigated by calculating speedup and efficiency ratings based on the computed aggregate megaflop rate as the problem size and the number of processors increase. Assume that the flop count to compute the Schur decomposition is $18N^3$, where N is the order of the matrix [3].

Let *efficiency* with respect to megaflop rate be defined as

$$E_F = \frac{M_p}{PM_s}$$

where M_p is the megaflop rate on P processors and M_s is the serial megaflop rate. The *speedup* with respect to megaflop rate

$$S_F = PE_F$$

is the factor by which execution time is reduced on P processors.

In Table 2, the time in seconds to compute the Schur decomposition for increasing larger problems and larger processor grids is shown. Notice that the efficiency ratings are much higher for the processor grids that are evenly divisible by four in Table 2. In fact, the megaflop rate per processor for the processor grids divisible by four stays approximately constant around 47. The IBM Power3 SMP currently has a maximum of 4 MPI processes per node, and intranode messages are communicated significantly faster than internode messages. For processor grids not evenly divisible by four, the batch system must be set to either use less MPI processes per node to balance the number of processes across nodes or to use four MPI processes per node with one node using less than four MPI processes. For example, with a 3×3 processor grid, an efficiency of 0.73 is obtained if the machine is set to use three MPI processes on three nodes instead of a four-four-one setup on three nodes that had an efficiency of 0.57 (as shown in Table 2). Thus, the variable efficiencies are a by-product of the hardware, not the code.

Table 2. *Speedups and efficiencies based on the megaflop rate for the IBM Power3 SMP.*

Speedups and efficiencies for IBM Power3 SMP						
Proc. grid		IBM Power3 SMP				
mp × np	N	Time	Mflops/s	S_F	E_F
1 × 1	2000	2577	55.9	1.0	1.00
2 × 2	4000	5975	192.8	3.5	0.86
3 × 3	6000	13550	286.9	5.1	0.57
4 × 4	8000	12526	735.8	13.2	0.82
5 × 5	10000	23712	759.1	13.6	0.54
6 × 6	12000	18078	1720.4	30.8	0.85
8 × 8	16000	24471	3012.9	53.9	0.84

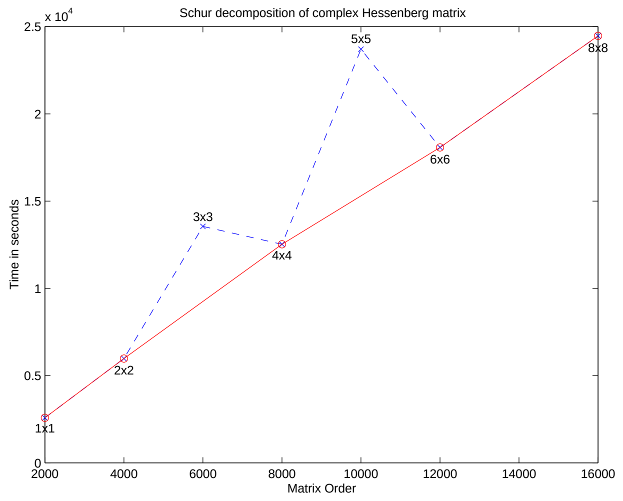


Figure 1. *Time in seconds to compute Schur decomposition of a complex Hessenberg matrix on an IBM Power3 SMP using increasing larger processor grids as the order of the matrix increases.*

The data in Table 2 are also displayed in Figure 5.1 and clearly show linear scalability for processor grids divisible by four.

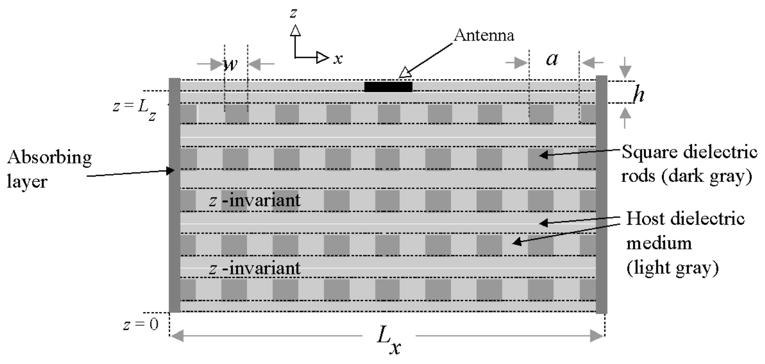
5.2 Electromagnetics case study

Properties of electronic components are intimately related to the behavior of electrons in a periodic crystalline structure. Similarly, photons propagating in a periodic structure can exhibit behavior analogous to an electron propagating in an electronic crystal. Typically, there can be one or more gaps in the range of allowable energy an electron propagating in a semiconductor crystal can have. Analogous band gaps can occur for the allowable energies of photons propagating in a periodic dielectric structure, i.e., photonic crystal (PC). The rest of this subsection summarizes the work in [7] where numerical calculation of radiation from a long antenna situated on a photonic crystal substrate was studied. It is this application where the new parallel eigenvalue solvers were first applied.

Theoretical background

The theory portion of this work involves solving Maxwell's equations using perfectly matched layer (PML) boundary conditions, the R-matrix propagator algorithm, and a finite-difference frequency-domain modal-expansion approach to calculate antenna radiation. The antenna is mounted adjacent to a finite-sized PC substrate and is driven at frequencies below, within, and above the band gap associated with an infinite PC. The PC and antenna are invariant along one dimension.

A photonic crystal (PC) and antenna structure is modeled as shown in Figure 5.2. The horizontal lines in Figure 5.2, separating regions of the PC, indicate



z
x
Antenna
w
a
h
z = L_z
Absorbing layer
Square dielectric rods (dark gray)
z -invariant
Host dielectric medium (light gray)
z -invariant
z = 0
L_x

Figure 2. *Photonic crystal and antenna structure.*

regions that are z invariant and are called layers henceforth. The problem analyzed here is two-dimensional, and the PC structure consists of a truncated square array of dielectric rods.

In the context of a finite-difference time-domain approach to solve Maxwell's equations, the analogous equations to Berenger's [2] work are equivalent to

$$\frac{\partial E_y(x,z)}{\partial z} = -\frac{i\omega}{c}\mu_z(x,z)B_x(x,z)$$

$$\frac{\partial B_x(x,z)}{\partial z} = -\frac{i\omega}{c}\epsilon_z(x,z)E_y(x,z) + \frac{\epsilon_z(x,z)}{\epsilon_x(x,z)}\frac{\partial}{\partial x}\left[\frac{c}{i\omega\mu_x(x,z)}\frac{\partial E_y(x,z)}{\partial x}\right] + \frac{4\pi}{c}J_y(x,z)$$

where

$$\epsilon_x(x,z) = \epsilon(x,z) + 4\pi i\sigma_x(x,z)/\omega, \qquad \mu_x(x,z) = 1 + 4\pi i\sigma_x^*(x,z)/\omega,$$
$$\epsilon_z(x,z) = \epsilon(x,z) + 4\pi i\sigma_z(x,z)/\omega, \qquad \mu_z(x,z) = 1 + 4\pi i\sigma_z^*(x,z)/\omega,$$

$J_y(x,z)$ is the antenna current, $E(x,z)$ is the electric field, and $B(x,z)$ is the magnetic field. Note that the antenna current is written as a function of x and z, but it is assumed the current is zero everywhere outside the antenna cross section and uniform within the cross section.

Assuming that material parameters (permittivity and PML conductivity) are independent of z, and noting that certain variables are zero outside of any PML region, the two equations can be combined into a second-order differential equation. Using centered finite-differences to approximate the x derivatives, there are N coupled differential equations for each z-invariant layer. All N equations for a given layer may be concisely written in matrix form as

$$\frac{\partial^2 E(z)}{\partial z^2} = ME(z) - 4i\pi\omega J/c^2 \tag{1}$$

where M is an $N \times N$ matrix. Explicit dependence on x has been omitted as well as y component notation in Equation (1). The solution for the fields in a layer is a straightforward diagonalization of M as $S^{-1}MS = \Lambda^2$, which yields the modal solutions

$$E(z) = S\left(\exp(\Lambda z)C_+ + \exp(-\Lambda z)C_-\right) + \frac{4i\pi\omega}{c^2}M^{-1}J \tag{2}$$

$$B(z) = \left(\frac{ic}{\omega}\right)S\Lambda(\exp(\Lambda z)C_+ - \exp(-\Lambda z)C_-) \tag{3}$$

where $C_\pm$ are vectors of constants. Clearly, N eigenvalues and eigenvectors must be computed for *each* layer of the PC. The root eigenvalues consist of pairs $\pm\Lambda$ where the $+\Lambda$ and $-\Lambda$ are associated with solutions which are "upward" or "downward" propagating within a layer.

Equations (2) and (3) only yield solutions within a layer. The R-matrix algorithm provides a numerically stable algorithm to give a relationship between field solutions across one or more layer boundaries [8, 10, 9]. The numerics of this problem include linear equation solutions, matrix-matrix multiplies, matrix-vector multiplies, and calculating eigenvectors and eigenvalues. Most of the computational effort involves repetitious calculation of eigenvectors and eigenvalues.

Scalability tests

Consider the PC shown in Figure 5.2. All material media are assumed to be non-dispersive for all frequencies with the permittivity of the dielectric rods $\epsilon_a = (9, 0)$, the background medium $\epsilon_b = (1, 0)$, and the wire antenna $\epsilon_w = (-100, 30)$. The superstrate and substrate regions above and below the PC have permittivity $(1, 0)$. The side dimension of the square rods is given by the fill factor of 0.156 or $w = 0.395a$. The PC consists of 37 periods in the x direction or $L_x = 37a$ and 6 whole periods in the z direction or $L_z = 6a$. The dimensions of the rectangular antenna are width $0.5a$ and height $0.2a$.

In the numerical results, the number of digitization points for the x dimension is $N = 701$ or 1401, which yields a spatial resolution of $\Delta x = 37a/N = 0.053a$ or $0.0265a$. The PML region on each side of the PC consists of 30 layers for a total thickness of $30\Delta x$. Within these layers, $\sigma_x(x) = \epsilon(x)\sigma_x^*(x)$, and the conductivity is quadratically increased with depth into the PML layers region.

Table 3 shows the results of parallelizing the electromagnetics code with ScaLAPACK and with the newly developed eigenvalue and eigenvector solvers. As

Table 3. *Observed wallclock times for the electromagnetics application.*

Execution time in hours				
	Number of processors			
N	1	4	9	16
701	1.5	0.4	0.2	-
1401	> 10	2.7	1.5	0.9

can be seen in Table 3, the parallelized code has delivered greatly reduced wallclock times for this application. For 4 processors, an efficiency of about 0.9 is observed, while for larger numbers of processors an efficiency of approximately 0.7 is observed.

5.3 Magnetohydrodynamics case study

Many important magnetohydrodynamic (MHD) systems contain shear in both the velocity and magnetic field. An important situation occurs when both shears are in the same location in a slab geometry [4, 5] where this configuration is termed the plane current-vortex sheet.

Most of the theoretical research on the plane current-vortex sheet has been concerned with the incompressible problem. The assumption of incompressibility is extremely useful for making the problem more tractable. This work pertains to an investigation into the compressible plane current-vortex sheet. The compressible analog of the well-studied incompressible problem is treated with a spatially uniform zeroth-order mass density and temperature, stressing how variations in the sonic Mach number (M) change the stability properties. The equations for the problem are briefly developed to show the underlying numerical calculations in the code. This is followed by a short description of the parallel implementation and then results of a scalability analysis.

Theoretical background

The following theoretical derivation comes from the work in [6] where the compressible plane current-vortex sheet was studied. The nonlinear partial differential equations that govern the behavior of a three-dimensional, compressible, dissipative magnetofluid in a dimensionless form are as follows

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho v), \tag{4}$$

$$\rho \left(\frac{\partial v}{\partial t} + v \cdot \nabla v \right) = -\frac{1}{\gamma M^2} \nabla P + A^2 J \times B + \frac{1}{R} \nabla \cdot \zeta, \tag{5}$$

$$\begin{aligned}\rho \left(\frac{\partial T}{\partial t} + v \cdot \nabla T \right) &= \frac{\gamma - 1}{\gamma} \left(\frac{\partial P}{\partial t} + v \cdot \nabla P \right) + \frac{1}{R\mathrm{Pr}} \nabla \cdot (\mu(T) \nabla T) \\ &\quad + \frac{M^2}{R} (\gamma - 1) \zeta_{ij} e_{ij} + \frac{A^2 M^2}{R_m} (\gamma - 1) \eta(T) (\nabla \times B)^2,\end{aligned} \tag{6}$$

$$\frac{\partial B}{\partial t} = \nabla \times v \times B - \frac{1}{R_m} \nabla \times \eta(T) \nabla \times B, \tag{7}$$

$$\nabla \cdot B = 0 \tag{8}$$

where $\rho(x,t)$ is the mass density, $v(x,t) = (u,v,w)$ is the flow velocity, $P(x,t)$ is the thermal pressure, $B(x,t) = (B_x, B_y, B_z)$ is the magnetic induction field, $T(x,t)$ is the plasma temperature, $\zeta_{ij} = \mu e_{ij} + (\lambda e_{kk} - P)$, $e_{ij} = \frac{1}{2}(\partial_j v_i + \partial_i v_j)$ is the viscous stress tensor, and γ is the adiabatic ratio.

The Equations (4)-(8) can be linearized as shown in [6], but will not be shown here for brevity. To solve the resulting equations, the Spectral Compressible Linear Stability (`SPECLS`) code [13], a Chebyshev collocation code, was modified to include magnetic effects. The modified version is denoted as `MHDLA`. The most time-consuming task in `MHDLA` is the computation of eigenvalues of a complex dense matrix. For serial runs of this code, more than 90% of the wallclock time is spent computing eigenvalues. Thus, it was apparent that the new parallel eigenvalue solver `PZLAHQR` could be used to significantly reduce wallclock times.

However, the entire code was not parallelized, only the routine where the eigenvalue calculations are performed. Although perfect linear speedup cannot be expected, use of the new parallel solver is expected to dramatically reduce wallclock times. Additionally, a second routine with some basic linear algebra tasks will be parallelized in the future.

Scalability tests

An example of a physical application is the formation and acceleration of the slow solar wind where the incipient wind is modeled as a wake and the accompanying coronal streamer magnetic field is modeled as a simple sheared magnetic field. The numerical tests were obtained using the `MHDLA` code to study the compressible problem with combined flow and magnetic shears.

In Table 4, the results of running the `MHDLA` code for varying numbers of Chebyshev polynomials with the serial and parallelized code are shown.

Table 4. *Observed wallclock times for the MHDLA code.*

Execution time in minutes				
	Number of processors			
N	1	4	9	16
807	1.7	0.7	-	-
1607	12.2	4.3	2.9	2.5
3207	114.1	38.5	20.5	14.1

Table 4 shows that greatly reduced wallclock times can be achieved in this application by only parallelizing the eigenvalue solver. As expected, the parallel speedup is low. This is because there is another routine with linear algebra computations that needs to be parallelized whose wallclock time becomes significant as the size of the problem increases. Since this part of the code is serial, it limits the speedup that can be achieved and lowers the parallel efficiency as processors are added. This limitation is an instance of Amdahl's Law.

6 Concluding Remarks

A new parallel complex Schur decomposition routine `PZLAHQR` has been implemented based on the ScaLAPACK code `PDLAHQR`. Results were shown for `PZLAHQR` and showed that this routine scales nicely with the number of processors. Several auxiliary subroutines were developed to support this routine that will be useful outside the scope of `PZLAHQR`.

In addition, a parallel eigenvector calculation routine `PZTREVC` was developed for complex upper triangular matrices. Also, a new version of `PZLATRS` was developed that uses scaling to control potential overflow. However, `PZLATRS` requires further work to improve its scalability.

Two application codes were also parallelized using ScaLAPACK and the new solvers presented here. In each case, significantly decreased wallclock times were observed. Since each code was relatively small on the order of 2000 lines, it only took a couple of days worth of work to implement the parallelization. This time will be recouped by the savings in wallclock time many times over as the researchers do much larger problems.

Acknowledgments

Computer runs were done on the SGI Origin 2000 and IBM Power3 SMP at the ERDC MSRC. This work was supported in part by a grant of computer time from the DoD High Performance Computing Modernization Program at the ERDC MSRC, Vicksburg, MS.

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide (second edition)*. SIAM, Philadelphia, PA, 1995.

[2] Jean-Pierre Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *J. Comp. Physics*, 127:363–379, 1996.

[3] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, 1997.

[4] R. B. Dahlburg. On the nonlinear mechanics of magnetohydrodynamic stability. *Phys. Plasmas*, 5:133–139, 1998.

[5] R. B. Dahlburg, P. Boncinelli, and G. Einaudi. The evolution of plane current-vortex sheets. *Phys. Plasmas*, 4:1213–1226, 1997.

[6] R. B. Dahlburg and G. Einaudi. The compressible plane current-vortex sheet. *Phys. Plasmas*, 7:1356–1365, 2000.

[7] J. Merle Elson and Mark R. Fahey. Numerical calculation of radiation from a long antenna situated on a photonic crystal substrate. In *Proceedings of the 1999 DoD User's Group Conference*, 1999.

[8] J.M. Elson and P. Tran. Dispersion and diffraction in photonic media: A different modal expansion for the R-matrix propagation technique. *J. Opt. Soc. Am. A*, 12:1765–1771, 1995.

[9] J.M. Elson and P. Tran. Band structure and transmission of photonic media: a real space finite-difference calculation with the R-matrix propagator. In C.M. Soukoulis, editor, *Photonic Band Gap Materials*, volume 315 of *NATO Advanced Study Institute Series E: Applied Sciences*, pages 341–354. Kluwer, Dordrecht, 1996.

[10] J.M. Elson and P. Tran. Couple mode calculation with the R-matrix propagator for the dispersion of surface waves on a truncated photonic crystal. *Phys. Rev.* **B**, 54:1711–1715, 1996.

[11] Mark R. Fahey. Parallel complex eigenvalue and eigenvector routines. Submitted to *ACM Trans. Math. Soft.*, Oct. 2000.

[12] G. Henry, D. Watkins, and J. Dongarra. A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. Computer Sciences Dept. Technical Report CS-97-352, University of Tennessee, Knoxville, TN, March 1997. LAPACK Working Note #121.

[13] M. G. Macaraeg, C. L. Streett, and M. Y. Hussaini. A spectral collocation solution to the compressible stability eigenvalue problem. National Aeronautics and Space Administration Technical Paper 2858, National Aeronautics and Space Administration, Washington, D.C., December 1988.